Trabajando con archivos y directorios en Python

Alex Felipe
27/10/2020

En mi sistema de registro de productos, necesito crear una función que, desde un archivo CSV con datos de productos, pueda leer este archivo y separar todos los productos que contiene. Para representar un producto tenemos la siguiente clase:

```
class Producto(object):
  def __init__(self, nombre, valor):
    self.__nombre = nombre
    self.__valor = valor

  @property
  def nombre(self):
    return self.__nombre

  @nombre.setter
  def nombre(self, nombre):
    self.__nombre = nombre

  @property def valor(self):
    return self.__valor

  @valor.setter
```

```
  def valor(self, valor):
    self.__valor = valor

  def __repr__(self):
    return "nombre:%s valor:%s" % (self.__nombre, self.__valor)
```

Inicialmente tenemos el archivo `datos.csv` con el siguiente contenido:

```
nombre, valor camiseta, 25.0 chaqueta, 125.0 zapatillas, 80.0 pantalones corto
```

En mi sistema, leeré este archivo del directorio "`archivos/productos/`", por lo tanto, crearemos la variable `archivo` indicando la ruta y nombre del archivo leeremos:

```
archivo = 'archivo/productos/datos.csv'
```

Implementando la función de lectura del archivo

Ahora necesitamos crear una función que se encargará de leer estos productos a través de la variable `archivo`, así que crearemos la función `leer_productos()` pasando el `archivo` como parámetro:

```
def leer_productos(archivo):

archivo = 'archivo/productos/datos.csv'
```

Primero, necesitamos abrir el archivo CSV, es decir, usaremos la función `open` enviando la variable `archivo` y usando el parámetro 'rb' que indica lectura:

```
def leer_productos(archuivo):
   archivo_abierto = open(archivo, 'rb')
```

En Python, para leer un archivo CSV, podemos usar la función `reader` del módulo `csv`, sin embargo, necesitamos importar el módulo:

```
import csv

def leer_productos(archuivo):
  archivo_abierto = open(archivo, 'rb')
  return csv.reader(archivo_abierto,delimiter=',')
```

Observe que también agregamos el parámetro `delimiter` que ya separa cada información de un criterio, en este caso, cada información estará separada en el momento en que aparezca una coma, y además, estamos devolviendo la función `reader` que es exactamente el archivo leído.

Entonces hagamos una prueba, haremos la llamada a la función `leer_productos` y lo devolveremos a la variable `datos`:

```
import csv

def leer_productos(archivo):
  archivo_abierto = open(archivo, 'rb')
  return csv.reader(archivo_abierto,delimiter=',')

archivo = 'archivo/productos/datos.csv'

datos = leer_productos(archivo)
```

Ejecutando nuestro código obtenemos el siguiente resultado:

```
IOError: [Errno 2] No such file or directory: 'archivo/productos/datos.csv'
```

Comprobando la existencia del archivo

¡Ups! ¡No pudo encontrar mi archivo! ¿Por qué pasó esto? Veamos dentro del directorio donde está mi proyecto:

```
> ls datos.csv model.py verificando_directorio.py
```

De hecho, no existe tal directorio, entonces, ¿qué debemos hacer? Antes incluso de intentar obtener el archivo, ¡tenemos que comprobar si al menos existe! En otras palabras,

primero debemos verificar si la ruta que queremos encontrar existe, si no es así, ¡debemos crearla!

Entonces, nuestro primer paso es separar la variable `archivo` en dos, es decir, el `archivo` y la `ruta`:

```
ruta = 'archivo/productos'
archivo = ruta + '/datos.csv'
```

A continuación, creemos la función `verificar_archvo`, que se encargará de verificar si el directorio existe y el archivo también, si no existe, ¡debe crearlos! Así que vamos a crearlo:

```
import csv

def verificar_archivo():
  ruta = 'archivo/productos' archivo = ruta + '/datos.csv'

def leer_productos(archivo):
  archivo_abierto = open(archivo, 'rb')
  return csv.reader(archivo_abierto,delimiter=',')

datos = leer_productos(archivo)
```

Comprobando la existencia del directorio

¿Cuál es nuestro próximo paso? Es precisamente para comprobar que el directorio no existe. Para eso, en Python, podemos usar el módulo os con funciones capaces de realizar llamadas de sistema.

En nuestro primer caso, usaremos la función `path.exists` que comprueba si existe un archivo o directorio con el parámetro especificado:

```
import csv import os

def verificar_archivo():
  ruta = 'archivo/productos'
  archivo = ruta + '/datos.csv'
```

```
if not os.path.exists(ruta):
```

Tenga en cuenta que estamos comprobando si el directorio no existe, si esto es cierto, ¿que debemos hacer? ¡Necesitamos crear el directorio! Pero, ¿cómo creamos un directorio en Python? ¡Sencillo! En módulo `os`, también tenemos la función `makedirs` que crea directorios:

```
def verificar_archivo():
  ruta = 'archivo/productos'
  archivo = camino + '/datos.csv'

if not os.path.exists(ruta):
   os.makedirs(ruta)
```

Nuestro paso siguiente es verificar la existencia del archivo dentro de este directorio. Podemos usar nuevamente la función `path.exists` del módulo `os`. De la misma manera que hicimos con el directorio, si el archivo no existe, necesitamos crearlo.

Crear el archivo

Para crear un archivo usaremos la función open pasando el segundo parámetro con el valor "w" que indica la creación para escrita:

```
def verificar_archivo():
  ruta = 'archivo/productos'
  archivo = ruta + '/datos.csv'

if not os.path.exists(ruta):
  os.makedirs(ruta)
if not os.path.exists(ruta):
  open(archivo, 'w')
```

Finalmente, devolvemos la variable `archivo` para leerla en la función `leer_productos`:

```
def verificar_archivo():
  ruta = 'archivo/productos'
  archivo = ruta + '/datos.csv'

  if not os.path.exists(ruta):
    os.makedirs(ruta)

  if not os.path.exists(archivo):
    open(archivo, 'w')

  return archivo
```

Ahora, basta que llamemos la función `verificar_archivo` devolviendo su resultado para la variable `archivo`, y luego, llamamos la función `leer_productos` enviando la variable `archivo` como parámetro:

```
arquivo = verificar_archivo() datos = leer_productos(archivo)
```

Exhibiendo las informaciones

Ahora vamos a imprimir los datos para comprobar los valores, sin embargo, en lugar de hacer un `for` procedimentalmente, utilizaremos el recurso de comprensión de lista:

```
print [dato for dato in datos]
```

Ejecutando el código obtenemos el siguiente resultado:

```
[['nombre', ' valor'],
 ['camiseta', ' 25.0'],
 ['chaqueta', ' 125.0'],
 ['zapatillas', ' 80.0'],
 ['pantalones cortos', ' 40.0']]
```

Tenga en cuenta que el primer valor sigue siendo el encabezado, es decir, las líneas 'nombre' y 'valor'. Para eliminar este encabezado simplemente llame la función `next` antes de iterar la variable `datos`:

```
next(datos) print [dato for dato in datos]
```

Finalmente, solo necesitamos almacenar los valores en objetos del tipo `Producto`:

```
productos = [Producto(dato[0], dato[1]) for dato in datos] print productos
```

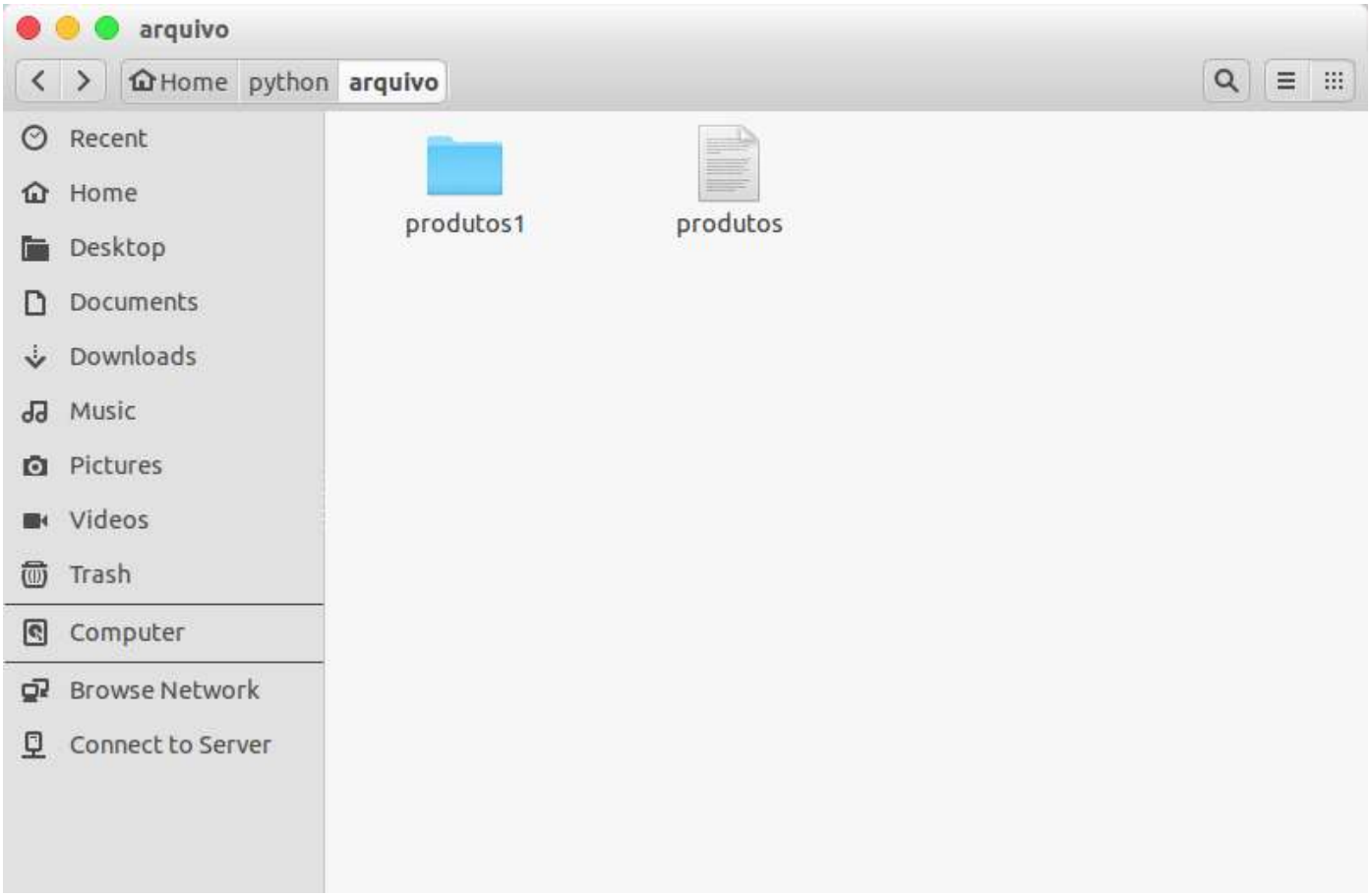
Ejecutando el código nuevamente, obtenemos el siguiente resultado:

```
[nombre:camiseta valor: 25.0,
nombre:chaqueta valor: 125.0,
nombre:zapatillas valor: 80.0,
nombre:pantalonescortos valor: 40.0]
```

Situaciones inesperadas

¡Tenga en cuenta que todos los datos se han leído sin ningún problema! Sin embargo, si la ruta "`archivo/productos`" en lugar de ser un directorio, ¿es un archivo? ¿Qué pasaría con nuestro código?

¡Hagamos una simulación! Primero voy a cambiar el nombre del directorio "`archivo/productos`" a "`archivo/productos1`" y luego, voy a crear un archivo llamado "productos" dentro del directorio "archivo":



Probando nuestro código, tenemos el siguiente resultado:

```
Traceback (most recent call last): File "/home/alex-felipe/python/dir.py", lin

 File "/home/alex-felipe/python/dir.py", line 14, in verificar_archivo open(ar
```

Vea que intentó abrir el archivo con la función `open`, pero, se dio cuenta de que el "`archivo/productos`" ¡es un archivo! En otras palabras, permitimos que nuestro algoritmo siguiera adelante ... Entonces, ¿qué debemos hacer?

Antes de que nuestro algoritmo intente abrir el archivo, ¡debemos asegurarnos de que la ruta sea válida! Entonces, justo cuando verificamos si no existe la ruta:

```
if not os.path.exists(ruta): os.makedirs(ruta)
```

Podemos agregar un `elif` que se activará si hay un archivo o directorio de acuerdo con la ruta especificada, y luego, verificará si la ruta no es un directorio:

```
if not os.path.exists(ruta): os.makedirs(ruta) elif not os.path.isdir(ruta):
```

Si es cierto, es decir, la ruta especificada no es un directorio, simplemente arrojamos un error de `IOError` advirtiendo que la ruta que estamos usando no es un directorio:

```
if not os.path.exists(ruta): os.makedirs(ruta) elif not os.path.isdir(ruta): r
```

Ahora, cuando existe un archivo o directorio en la ruta especificada **y no es un directorio**, ¡detenemos automáticamente nuestro algoritmo! Veamos el resultado de la prueba:

```
Traceback (most recent call last): File "/home/alex-felipe/python/dir.py", lin
```

Vea que ahora, incluso en casos excepcionales como este, nuestro algoritmo puede manejar y tomar las decisiones necesarias.

Conclusión

En esta publicación vimos los problemas que podemos tener al leer archivos dentro de directorios, es decir, antes de tener que leer un archivo **siempre necesitamos** verificar si la ruta, en este caso, el directorio y el archivo, realmente existen, si no, necesitamos crearlos, si sí, ¡simplemente leemos!

¿Qué tal aprender más sobre **Python** y sus diversos recursos? Entonces, ¡Mira nuestros cursos de **Python para Data Science** aquí en Alura!

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN

DEVOPS

AOVS Sistemas de Informática S.A
CNPJ 05.555.382/0001-33

SÍGUENOS EN NUESTRAS REDES SOCIALES

ALIADOS

Empresa participante do SCALE UP ENDEAVOR

En Alura somos unas de las Scale-Ups seleccionadas por Endeavor, programa de aceleración de las empresas que más crecen en el país.

Google for Startups Alumni 2021

Fuimos unas de las 7 startups seleccionadas por Google For Startups en participar del programa Growth Academy en 2021

POWERED BY

CURSOS

Cursos de Programación
Lógica de Programación | Java

Cursos de Front End
HTML y CSS | JavaScript | React

Cursos de Data Science

Data Science | Machine Learning | Excel | Base de Datos | Data Visualization | Estadística

Cursos de DevOps

Docker | Linux

Cursos de Innovación y Gestión

Productividad y Calidad de Vida | Transformación Ágil | Marketing Analytics | Liderazgo y Gestión de Equipos | Startups y Emprendimiento